**Fake news detection/analysis**

**A Project Report submitted the fulfilling the requirement**

**for the degree of**

**INFORMATIONAL TECHNOLOGY AND MANAGEMENT**

**UNDER**

**UTKAL UNIVERSITY,**

**BHUBANESWAR, ODISHA**

**Submitted by: Guided by:**

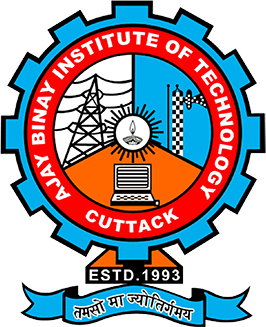
DR. Amaresh Sahu

Evita Bhowmik

Bijoy Laxmi Behera

Elina Priyadarsini Mohanty

Sasmita Jena



**AJAY BINAYA DEGREE COLLEGE**

**CUTTACK, ODISHA**

**Abstract**

1. **Objective:**

The objective of fake news analysis is to distinguish between false or misleading information and legitimate news stories, by examining the credibility and accuracy of the information presented. Fake news can be intentionally created and spread to deceive people, manipulate public opinion, or achieve other malicious purposes. Analyzing fake news involves investigating the source of the information, the language used, and the veracity of the claims made, as well as considering the context and potential biases of the author or publisher. The goal of this analysis is to help people make informed decisions about what information to believe and to prevent the spread of misinformation.

1. **Introduction:**

Fake news analysis is the process of identifying and analyzing false information that is disseminated with the intent to mislead or deceive people. Fake news can be spread through various mediums such as social media, websites, and traditional news channels.

The analysis of fake news typically involves using various techniques from data science and natural language processing to identify patterns and trends in the spread of false information. These techniques include sentiment analysis, network analysis, and classification algorithms.

The importance of fake news analysis has grown significantly in recent years, as the proliferation of false information has become a major concern for governments, media organizations, and individuals alike. Fake news can have serious consequences, such as influencing public opinion, distorting facts, and even inciting violence.

By analyzing fake news, researchers and analysts can better understand the motivations behind the spread of false information, as well as develop strategies for combating it. This can involve creating tools and techniques for fact-checking and verifying information, as well as educating the public on how to identify and avoid false information.

1. **Prerequisites:**

the necessary things requirements to run our program are as follows:

* Install VS Code
* Install Jupiter notebook
* Install the necessary libraries required

1. **Installation:**

* **Install VS Code:**

Go to <https://code.visualstudio.com/download> and download VS Code according to your PC/Laptop.

* **Install Jupiter notebook:**

To download the Jupyter Notebook extension in VS Code, follow these steps:

* Open VS Code and click on the Extensions icon on the left-hand side of the screen (or use the keyboard shortcut **Ctrl + Shift + X**).
* In the search bar, type "Jupyter" and press Enter.
* Look for the "Jupyter" extension and click on the "Install" button.
* Wait for the installation to finish and then click on the "Reload" button to activate the extension.

Once the Jupyter extension is installed, you can create and open Jupyter notebooks in VS Code.

* **Install the required python libraries :**

open the terminal and run the following commands to install the required python libraries.

* pip install numpy
* pip install pandas
* pip install scikit-learn
* pip install superml
* pip install sklearn-features
* pip install tfidf
* pip install spoke-scikit-learn
* pip install scikit-learn
* pip install datacamprojects
* **Run the notebook:**

To run the project, open VS Code and navigate the project directory . then run the program in Jupiter notebook.

1. **Why are we using Jupiter notebook?**

Jupyter Notebook is a popular tool used for analysing data and developing code in various programming languages, including Python, which is commonly used in data analysis.

Fake news analysis involves processing large amounts of data, such as news articles and social media posts, to identify patterns and trends that can help determine the veracity of the content. Jupyter Notebook allows analysts to write and execute code in a flexible and interactive environment, which is especially useful for data analysis tasks.

Jupyter Notebook also allows for the easy sharing of code and results with others, making it a popular choice for collaborative projects. Its support for visualizations, such as graphs and charts, can also aid in the interpretation of data.

Overall, Jupyter Notebook is a powerful tool for analysing and interpreting data, which makes it a useful tool for fake news analysis.

1. **Database:**

we have taken a CSV file named “news.csv” which contains 4 fields. 1st one contains some kind of code number, 2nd one contains the title which basically contains headlines, 3rd one contains text which contains brief news regarding that headline, and 4th one contains the level which shows the news is ‘fake’ or ‘Real’.

Here, I have attached the CSV file (to open the file press the ctrl key and click on the below icon).

[](news.csv)

1. **Project structure:**
2. import numpy as np
3. import pandas as pd
4. import itertools
5. from sklearn.model\_selection import train\_test\_split
6. from sklearn.feature\_extraction.text import TfidfVectorizer
7. from sklearn.linear\_model import PassiveAggressiveClassifier
8. from sklearn.metrics import accuracy\_score, confusion\_matrix

#Read the data

df=pd.read\_csv('news.csv')

#Get shape and head

df.shape

df.head()

#DataFlair - Get the labels

labels = df['label']

labels.head()

#DataFlair - Split the dataset

# Split the dataset

x\_train, x\_test, y\_train, y\_test = train\_test\_split(df['text'], df['label'], test\_size=0.2, random\_state=7)

#DataFlair - Initialize a TfidfVectorizer

tfidf\_vectorizer=TfidfVectorizer(stop\_words='english', max\_df=0.7)

#DataFlair - Fit and transform train set, transform test set

tfidf\_train=tfidf\_vectorizer.fit\_transform(x\_train)

tfidf\_test=tfidf\_vectorizer.transform(x\_test)

#DataFlair - Initialize a PassiveAggressiveClassifier

pac=PassiveAggressiveClassifier(max\_iter=50)

pac.fit(tfidf\_train,y\_train)

#DataFlair - Predict on the test set and calculate accuracy

y\_pred=pac.predict(tfidf\_test)

score=accuracy\_score(y\_test,y\_pred)

print(f'Accuracy: {round(score\*100,2)}%')

#DataFlair - Build confusion matrix

confusion\_matrix(y\_test,y\_pred, labels=['FAKE','REAL'])

1. **Code explanation:**

import numpy as np

import pandas as pd

import itertools

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.linear\_model import PassiveAggressiveClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix

* **numpy** (as **np**): a library for numerical computing in Python.
* **pandas** (as **pd**): a library for data manipulation and analysis.
* **itertools**: a module for efficient looping and iteration.
* **sklearn** (scikit-learn): a machine learning library for Python that provides tools for data preprocessing, classification, regression, clustering, and more.
  + **train\_test\_split**: a function for splitting data into training and testing sets.
  + **TfidfVectorizer**: a class for converting text data into a matrix of TF-IDF features.
  + **PassiveAggressiveClassifier**: a class for implementing the Passive-Aggressive online learning algorithm for binary classification.
  + **accuracy\_score**: a function for computing the accuracy of a classification model.
  + **confusion\_matrix**: a function for computing the confusion matrix of a classification model.

#Read the data

df=pd.read\_csv('news.csv')

#Get shape and head

df.shape

df.head()

The first line of the code imports the **pandas** library and uses the **read\_csv()** function to read a CSV file called "news.csv" into a **DataFrame** object called **df**.

The second line of the code uses the **shape** attribute of the **DataFrame** object to return the number of rows and columns in the dataset.

The third line of the code uses the **head()** method of the **DataFrame** object to display the first five rows of the dataset.

#DataFlair - Get the labels

labels=df.label

labels.head()

The code extracts the **label** column from the **DataFrame** object **df** and assigns it to a new variable **labels**.

The second line of code uses the **head()** method to display the first five rows of the **labels** variable.

#DataFlair - Split the dataset

# Split the dataset

x\_train, x\_test, y\_train, y\_test = train\_test\_split(df['text'], df['label'], test\_size=0.2, random\_state=7)

This code is splitting the dataset into two parts:

1. **x\_train** and **y\_train**: This is the training set, which the model will use to learn patterns and relationships between the input features (i.e., the text of the news articles) and the corresponding labels (i.e., whether the news is real or fake). **x\_train** is a pandas Series object containing the text of the news articles, and **y\_train** is a pandas Series object containing the labels.
2. **x\_test** and **y\_test**: This is the testing set, which the model will use to evaluate how well it has learned to classify news articles as real or fake. **x\_test** is a pandas Series object containing the text of the news articles, and **y\_test** is a pandas Series object containing the labels.

The **train\_test\_split()** function from scikit-learn's **model\_selection** module is used to split the data. It takes four arguments:

* **df['text']**: This is the input data, which is a pandas Series object containing the text of the news articles.
* **df['label']**: This is the target variable, which is a pandas Series object containing the labels (i.e., whether the news articles are real or fake).
* **test\_size=0.2**: This parameter specifies the proportion of the dataset that should be used for testing. Here, 20% of the data will be used for testing.
* **random\_state=7**: This parameter sets the random seed for reproducibility. The same seed value will produce the same train-test split each time the code is run.

The resulting variables (**x\_train**, **x\_test**, **y\_train**, and **y\_test**) will be used to train and evaluate a text classification model using scikit-learn's **TfidfVectorizer** and **PassiveAggressiveClassifier** classes.

#DataFlair - Initialize a TfidfVectorizer

tfidf\_vectorizer=TfidfVectorizer(stop\_words='english', max\_df=0.7)

#DataFlair - Fit and transform train set, transform test set

tfidf\_train=tfidf\_vectorizer.fit\_transform(x\_train)

tfidf\_test=tfidf\_vectorizer.transform(x\_test)

This code block is used for initializing and applying a Term Frequency-Inverse Document Frequency (TF-IDF) vectorization on the text data.

TF-IDF is a numerical statistic used to reflect the importance of a word in a document within a collection or corpus of documents. The TF-IDF vectorization process takes the raw text data as input and transforms it into a numerical feature vector, where each feature represents a unique word in the text corpus.

In this code, we first initialize a **TfidfVectorizer** object with the following parameters:

* **stop\_words='english'**: This parameter removes common English words such as "a", "the", "and", etc., which do not have much predictive value.
* **max\_df=0.7**: This parameter removes words that occur in more than 70% of the documents, as these words are likely to be too common and not informative.

Next, we use the **fit\_transform** method of the **tfidf\_vectorizer** object to transform the training data **x\_train** into a TF-IDF matrix **tfidf\_train**. We fit and transform the training data together, which means that the vectorizer learns the vocabulary from the training data and uses it to transform both the training and test data.

Finally, we use the **transform** method of the **tfidf\_vectorizer** object to transform the test data **x\_test** into a TF-IDF matrix **tfidf\_test**.

#DataFlair - Initialize a PassiveAggressiveClassifier

pac=PassiveAggressiveClassifier(max\_iter=50)

pac.fit(tfidf\_train,y\_train)

#DataFlair - Predict on the test set and calculate accuracy

y\_pred=pac.predict(tfidf\_test)

score=accuracy\_score(y\_test,y\_pred)

print(f'Accuracy: {round(score\*100,2)}%')

This code implements a Passive Aggressive classifier, which is a type of online learning algorithm for binary classification tasks.

First, an instance of the PassiveAggressiveClassifier is created, with the maximum number of iterations set to 50. Then, the classifier is trained on the training data using the fit() method, where the feature vectors for the text in the training set are represented by the tfidf\_train object and the corresponding labels are in y\_train.

Next, the trained model is used to predict the labels for the test set using the predict() method, and the accuracy of the predicted labels is calculated using the accuracy\_score() method from scikit-learn. Finally, the accuracy score is printed to the console, rounded to 2 decimal places.

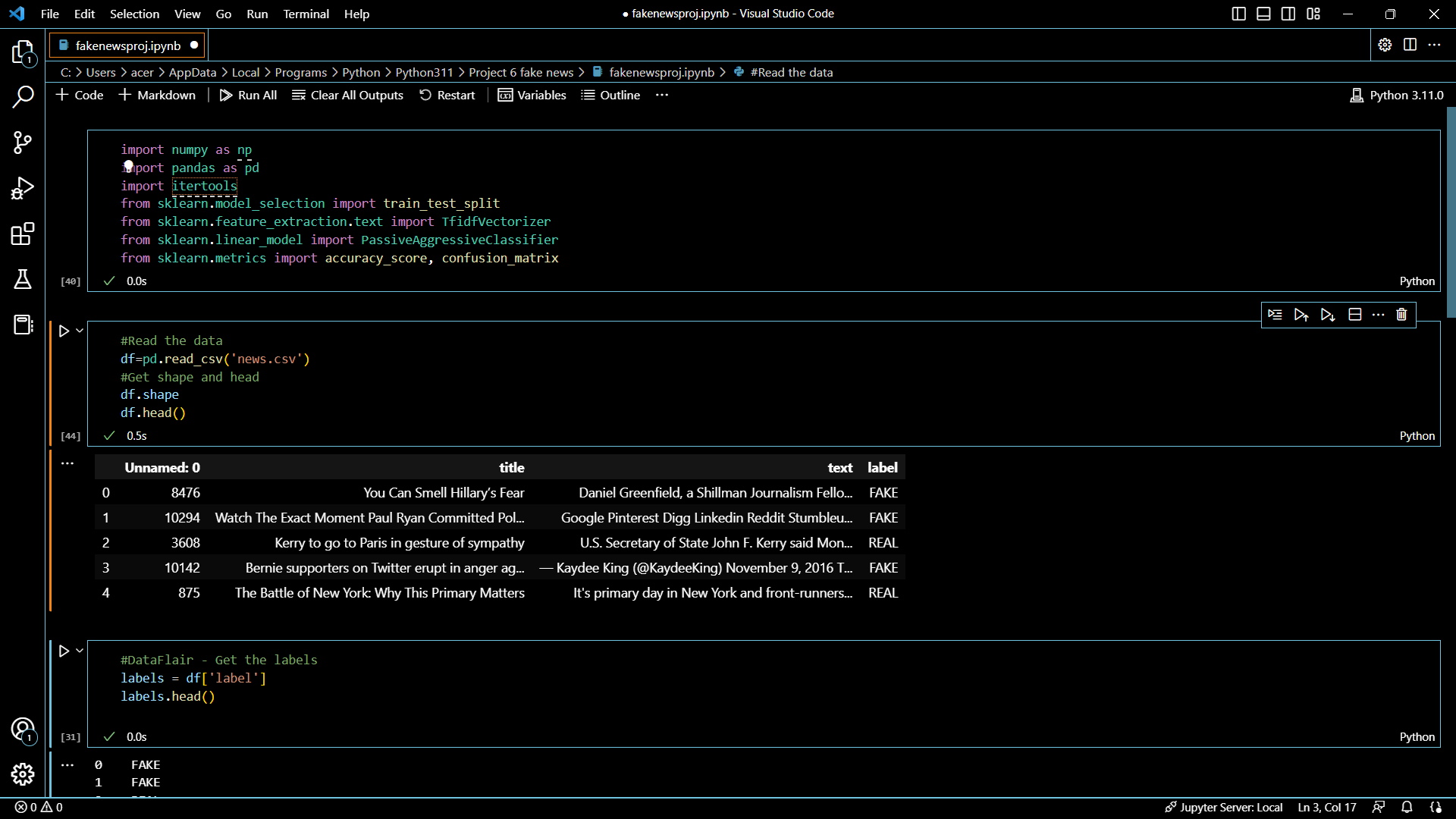
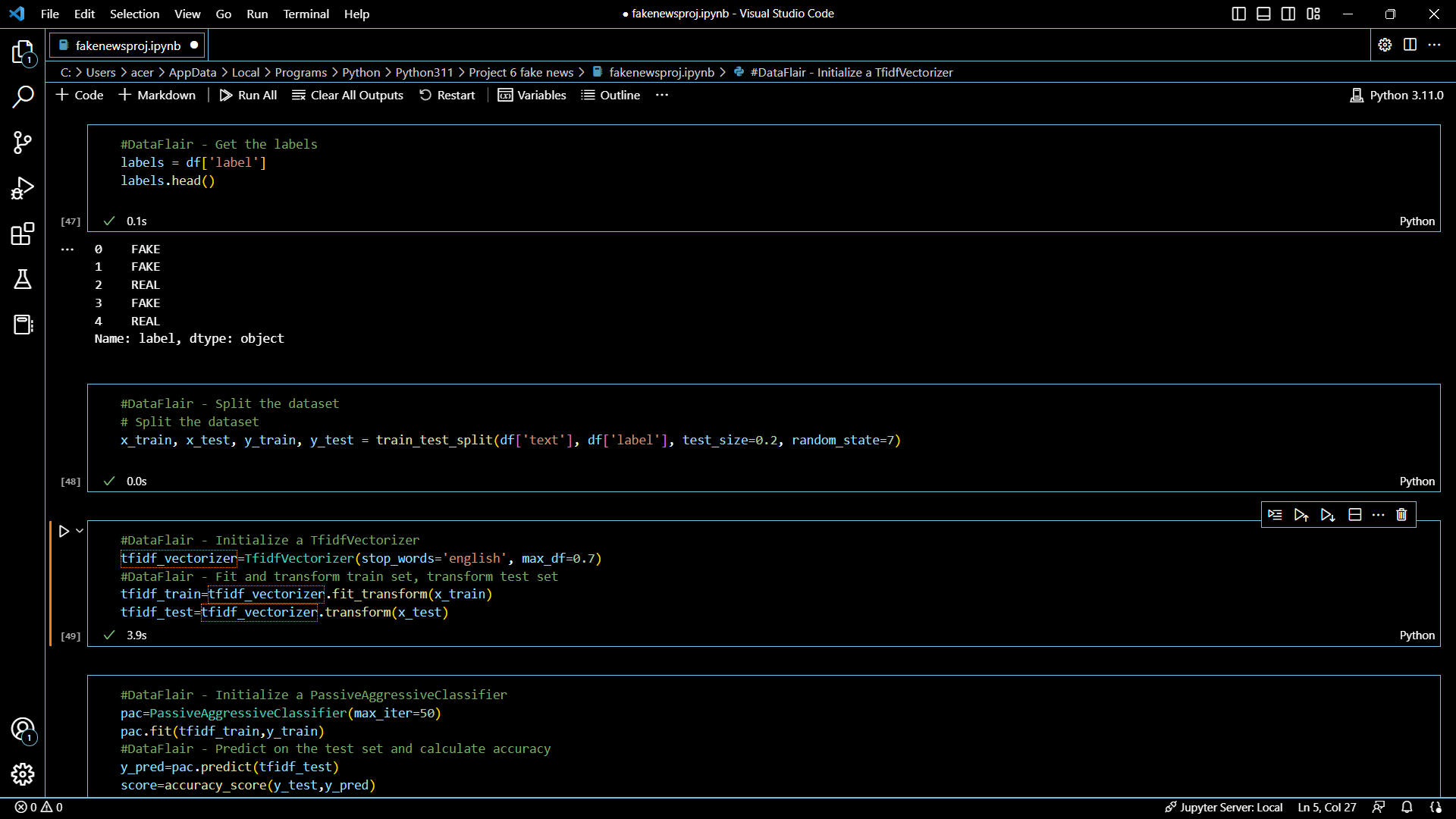
#DataFlair - Build confusion matrix

confusion\_matrix(y\_test,y\_pred, labels=['FAKE','REAL'])

This code calculates the confusion matrix for the predictions made by the model on the test set.

The confusion\_matrix() function from scikit-learn takes two arguments: the true labels (y\_test) and the predicted labels (y\_pred). The function also takes an optional parameter called "labels", which is a list of the class labels in the dataset. In this case, the labels are set to ['FAKE', 'REAL'], which are the two possible labels for the news articles.

The output of the confusion matrix is a 2x2 array that shows the number of true positives, false positives, true negatives, and false negatives for the classification task. The rows correspond to the true labels and the columns correspond to the predicted labels. The diagonal elements represent the correctly classified instances, while the off-diagonal elements represent the misclassified instances.

1. **Output:**

****The output of the code is the accuracy of the model and the confusion matrix.



1. **Conclusion:**

The detection of fake news is a complex and ongoing challenge, and there are no easy solutions. However, significant progress has been made in recent years with the development of various automated and human-based techniques for detecting fake news. While there is still room for improvement in the accuracy and effectiveness of these techniques, they have the potential to play a crucial role in mitigating the impact of fake news on society.

In the future, it is likely that fake news detection will become more sophisticated and comprehensive, incorporating a range of different techniques and technologies. There will also be a greater emphasis on education and media literacy to help individuals identify and avoid fake news.

Ultimately, the fight against fake news will require a collaborative effort from governments, technology companies, media organizations, and individuals. By working together, we can help to ensure that accurate information is prioritized and that the spread of fake news is minimized.